# Linear Regression

Code:

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

data=pd.read\_csv(r'iris\_data\_2a.csv')

print(data.head())

x=data.drop('species','columns')

y=data['species']

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.20)

#print(x\_train)

scaler=StandardScaler()

scaler.fit(x\_train)

#print(x\_train)

x\_train=scaler.transform(x\_train)

x\_test=scaler.transform(x\_test)

#print(x\_train)

#print(x\_test)

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(x\_train,y\_train)

result= regressor.predict(x\_test)

print("Result:\n",result)

# Evaluating Model's Performance

from sklearn.metrics import mean\_absolute\_error,mean\_squared\_error

print('Mean Absolute Error:', mean\_absolute\_error(y\_test, result))

print('Mean Squared Error:', mean\_squared\_error(y\_test, result))

print('Mean Root Squared Error:', np.sqrt(mean\_squared\_error(y\_test, result)))

Text

Description automatically generated with low confidence

# KMeanClustering

Code:

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

data=pd.read\_csv(r'iris\_data.csv')

x = data.iloc[:, :2].values

print(x)

#Finding the optimum number of clusters for k-means classification

from sklearn.cluster import KMeans

#Applying kmeans to the dataset / Creating the kmeans classifier

kmeans = KMeans(n\_clusters = 3, init = 'k-means++', max\_iter = 300, n\_init = 10, random\_state = 0)

y\_kmeans = kmeans.fit\_predict(x)

#Visualising the clusters

plt.scatter(x[y\_kmeans == 0, 0], x[y\_kmeans == 0, 1], s = 100, c = 'red', label = 'setosa')

plt.scatter(x[y\_kmeans == 1, 0], x[y\_kmeans == 1, 1], s = 100, c = 'blue', label = 'versicolor')

plt.scatter(x[y\_kmeans == 2, 0], x[y\_kmeans == 2, 1], s = 100, c = 'green', label = 'virginica')

#Plotting the centroids of the clusters

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:,1], s = 100, c = 'yellow', label = 'Centroids')

plt.legend()

plt.show()

